METHOD FOR DATA ACCESS CODE GENERATION

Field of the Invention

This invention relates in general to the field of computer software and more particularly to data access code generation utilizing templates.

Background of the Invention

The introduction of Java, an object-oriented, multi-threaded, portable, platform-independent, secure-programming environment, and its wide acceptance on the World Wide Web, requires programmers to migrate existing code and develop new code in Java.

In object-oriented programming such as Java, a class is a template definition of the methods and variables in a particular kind of object. For example, classes may be created for graphical user interface elements such as windows and tool bars and also for database interfaces. In general, classes provide an overall framework for developing an application program.

To address the requirements of commercial World Wide Web publishing and enable the expansion of World Wide Web technology, the World Wide Web Consortium has developed an Extensible Markup Language (XML). XML allows the creation of Java-based World Wide Web applications that were not previously possible.

Traditionally, Java classes have been manually created requiring the user to have knowledge of proprietary backend data access Application Program Interfaces (APIs). The manual generation of Java classes can often be complex and very time-consuming

resulting in inconsistent code generation. Furthermore, for different APIs, specific Java classes must be generated to provide access to the backend data.

Without a method and apparatus that improves the efficiency of generating data access Java classes, the computer industry will continue to be plagued by inconsistent implementations.

Summary of the Invention

According to the invention, there is provided a method and apparatus for data access code generation. A data access code generator routine applies code generation templates to an Extensible Markup Language (XML) document to generate the necessary data access Java classes.

The XML document describes a data object and the mapping of the data object to Java objects. The XML file must be created pursuant to the data model that exists in the backend Application Program Interface (API) as each backend API generally requires specific elements, attributes, and mappings to facilitate backend data access. The details of each data model are defined in a data object Data Type Definition (DTD) document.

The code generation templates contain the support for the backend API data access calls that are required to create the necessary Java classes from the XML file. The code generation template is implemented in any transformation language that can convert XML into Java code.

In operation, the data access code generator routine applies code generation templates to an XML document containing a data object description, which creates the resulting Java classes.

Other objects and advantages of the invention will become clear from the following detailed description of the preferred embodiment, which is presented by way of illustration only and without limiting the scope of the invention to the details thereof.

Brief Description of the Drawings

Further features and advantages will be apparent from the following detailed description, given by way of example, of a preferred embodiment taken in conjunction with the accompanying drawings, wherein:

Fig. 1 is a schematic block diagram of a data access code generator system constructed in accordance with the preferred embodiment of the present invention.

Detailed Description of the Invention

Throughout the figures, like elements are indicated by like reference numbers.

Referring to **Fig. 1**, a data access code generator **10** of the preferred embodiment is depicted having a data access code generator routine **18** and code generation templates **16**.

The data access code generator routine **18** takes the code generation templates **16** and the XML data object description **14** as inputs and produces the necessary data access Java classes **20**.

The XML data object description **14** is a document created pursuant to the data model that exists in the backend. The data model is defined in a data object Document Type Definition (DTD) document **12**. The data object DTD **12** contains all of the backend APIs definitions for the elements, attributes, and mappings that are required to

provide access to the backend data at run-time. For example, a system that accesses a relational database backend will contain attributes and elements for column names and the mapping of the columns to Java attributes.

Every object description contained within the XML data object description **14** must conform to the elements, attributes, and mappings defined in the data object DTD **12**. Further, the XML data object description **14** must be supported by the data access code generator routine **18**.

The code generation templates **16** describe the code that is generated when the data access code generation routine **18** is invoked. The code generator templates **16** contain the following elements:

data access API calls for the backends that are supported in the data object DTD;

the implementation of system-wide policies such as caching, lazy initialization, logging, and the use of system infrastructure; and

rules for code generation.

In particular, the incorporation of rules for code generation allows for the simple, straightforward, and consistent generation of Java classes.

The code generation templates **16** are implemented in any transformation language that can convert XML into Java code. For example, the code generation templates **16** can be implemented in a transform mechanism such as Extensible Stylesheet Language (XSL), in which case the code generation templates **16** are actually XML files.

In operation, the method and apparatus for data access code generation commences with an author creating an XML data object description **14** describing a data object. The description of the data object must conform to the elements, attributes, and mappings defined in the data object DTD **12**. The data access code generator routine **18** then takes the code generation templates **16** and the XML data object description **14** as inputs and produces the necessary data access Java classes **20** to provide access to the backend data at run-time.

One of the advantages of using a data access code generator routine **18** in conjunction with the code generation templates **16** is that programmers need not learn the proprietary backend APIs. The specific calls for each proprietary backend API are contained only in the code generation templates **16**, where they are applied to the XML data object description **14** to create the necessary data access Java classes **20**. This provides the further advantage of allowing the protection of proprietary information as programmers may be provided with the code generation templates **16** instead of the details of each proprietary backend API.

This method and apparatus also provide the advantage of allowing system-wide changes and enhancements to be made by simply regenerating the data access Java classes using modified code generation templates **16** or by simply modifying the data object description in the XML document **14**.

Furthermore, through simply modifying the code generation templates **16**, different policies can be applied without the need to edit each of the resulting data access Java classes **20**.

Therefore, this method and apparatus for data access code generation provides the advantages of modifiability, extensibility, adaptation, and reusability. The modifiability of the data object description in the XML document and the code generator templates

provides clear advantages over the manual creation and modification of Java classes. Consequently, a single XML data object description may be used in conjunction with several different code generation templates to create data access Java classes for different backend APIs. Furthermore, the code generation templates can be modified to include system modifications and the data access Java classes regenerated without the need to edit each resulting data access Java class.

Accordingly, while this invention has been described with reference to illustrative embodiments, this description is not to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, will be apparent to persons skilled in the art upon reference to this description.

For example, any object-oriented programming environment, such as C++, may be substituted for Java. Furthermore, a single code generation template may be used, or a plurality of code generation templates may be used. Similarly, the data access code generator routine may create a single data access Java class or it may create a plurality of data access Java classes.

It is therefore contemplated that the appended claims will cover any such modifications or embodiments as fall within the true scope of the invention.